51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2011年9月
总第006期

Java之父高斯林从谷歌离职

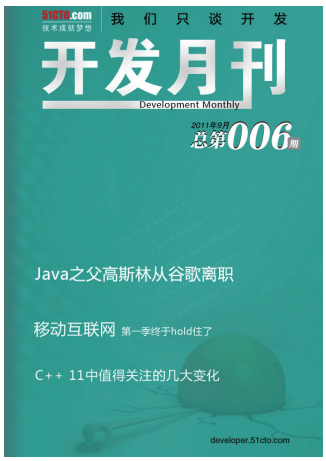
移动互联网 第一季终于hold住了

C++ 11中值得关注的几大变化

developer.51cto.com

目录

2011.09 contents



51CTO.com 技术成就梦想

出版方：北京无忧创想信息技术有限公司

责任编辑：彭凡 陈贻新 边艾婧 周少南 封面设计：魏峰

联系方法：pengfan@51cto.com 010-68476606（分机 8058）

出版日期：2011 年 9 月 15 日

9月排行榜:机器人设计语言NXT-G

最新的编程语言排行榜已经公布,与 8 月榜单相比,最引人瞩目的是 D 语言取代了 8 月份首次进入前 20 名的 F # 语言占居榜单第 20 位。另一个惊喜的是 Scala 重回前 50,虽然被称为下一代Java的JVM 语言 Scala 却一直未受 Android 开发者重视,但这会不会是一个契机呢?

与 8 月榜单相比,最引人瞩目的是 D 语言取代了 8 月份首次进入前 20 名的 F # 语言占居榜单第 20 位。和 F# 第一次进入 Top 20 不同,D 语言并非首次进入,它从 2007 年至 2009 年中期,一直在 Top 20。另一个惊喜的是 Scala 重回前 50,静态类型 Java 语言以 JRuby 及 Groovy 的后备队的身份,在移动 Android 应用程序开发领域整装待发。但是被称为下一代 Java 的 JVM 语言 Scala 却未受 Android 开发者重视,这会不会是一个契机呢?

下面是前 20 名的编程语言排行

Position Sep 2011	Position Sep 2010	Delta in Position	Programming Language	Ratings Sep 2011	Delta Sep 2010	Status
1	1	=	Java	18.761%	+0.85%	A
2	2	=	C	18.002%	+0.86%	A
3	3	=	C++	8.849%	-0.96%	A
4	6	↑↑	C#	6.819%	+1.80%	A
5	4	↓	PHP	6.596%	-1.77%	A
6	8	↑↑	Objective-C	6.158%	+2.79%	A
7	5	↓↓	(Visual) Basic	4.420%	-1.38%	A
8	7	↓	Python	4.000%	-0.58%	A
9	9	=	Perl	2.472%	+0.03%	A
10	11	↑	JavaScript	1.469%	-0.20%	A
11	10	↓	Ruby	1.434%	-0.47%	A
12	12	=	Delphi/Object Pascal	1.313%	-0.27%	A
13	24	↑↑↑↑↑↑↑↑↑↑	Lua	1.154%	+0.60%	A
14	13	↓	Lisp	1.043%	-0.04%	A
15	15	=	Transact-SQL	0.860%	+0.09%	A
16	14	↓↓	Pascal	0.845%	+0.06%	A-
17	20	↑↑↑	PL/SQL	0.720%	+0.08%	A--
18	19	↑	Ada	0.682%	+0.01%	B
19	17	↓↓	RPG (OS/400)	0.666%	-0.05%	B
20	30	↑↑↑↑↑↑↑↑↑↑	D	0.609%	+0.20%	B

8 月 30 日 Java 之父高斯林在自己的博客发表了一篇名为《我又跑路了》的博文。在博文里高斯林宣布自己将离开 Google,并解释称自己已经找到了更感兴趣的方向。高斯林新加盟的公司是 Liquid Robotics, Liquid Robotics 曾获得 2010 年度科技创新奖项中机器人技术奖。高斯林的加盟这也意味着 Liquid Robotics 开发出的利用潮汐能与太阳能来工作的机器人技术在未来将会更加智能。事实证明 Java 技术之父都对机器人技术感兴趣了,甚至不惜放弃 Google,那么您呢?本期我们就为大家介绍一种专为机器人玩具设计的可视化编程语言 NXT-G。

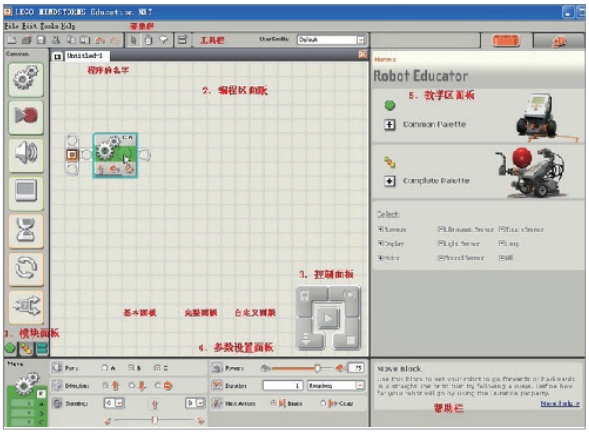
NXT-G 简介

NXT-G 是一种可视化的编程语言,它集成在乐高 (LEGO) 公司的机器人玩具产品中, NXT 程序的设计者可以在计算机上通过 NXT-G 对机器人的行为进行可视化的程序设计,并通过蓝牙等方式将这些指令下载到机器人身上,一旦 NXT 机器人程序化后,它就“开始有了自己的生命,不再需要经由计算机控制”。

NXT-G 编程语言的特殊之处在于其并非是提供给专业程序开发者的编程语言,而是由丹麦著名玩具制造商乐高公司为 NXT 机器人玩具设计的可视化编程语言,简称 G 语言。

9 月编程语言排行榜：机器人设计语言 NXT-G II

NXT-G 编程方法简单易学但不可小视，只要 c 语言、Java 语言编出的程序，NXT-G 都可以编出。乐高公司为这种编程语言开发的软件称为“LEGO MINDSTROMS NXT 编程 (programming)”，下文简称编程软件。



编程中的编程软件

编程软件分两种版本：玩具版本 (8547 NXT 零售版本附带) 和教育版本 (左图)(须在网上下载，只有 9797 NXT 教育版本用户可使用)，它们的主要区别是玩具版只有英语，而教育版本可选择语言。

编程模块与线程

NXT-G 语言最主要的编程方法则是用这些模块。每一个模块代表着文本编程语言的一个或多个语句。图中的模块从左自右依次是：“前进 (bc 马达)”“检测位于三端口的光电传感器”“转动 A 马达”。而压在模块下面的乐高横梁则代表线程，所以图中的程序是一个简单的单线程程序。模块可以任意从模块面板或编程区中的任意地方拉到横梁上并成为程序的一部分。当然，模块也可以从外部导入 (工具 -》模块导入导出向导 ...)，就像 c 中的外部头文件，还可以利用编辑 -》新建“我的模块”来新建自己的模块，就像 c 里的自己编写函数。而要分支出一个线程时，则需要用连接工具 (shift+ 鼠标点击) 把要连接的模块和主线程连接起来。

编程模块的设置与数据中心、数据线

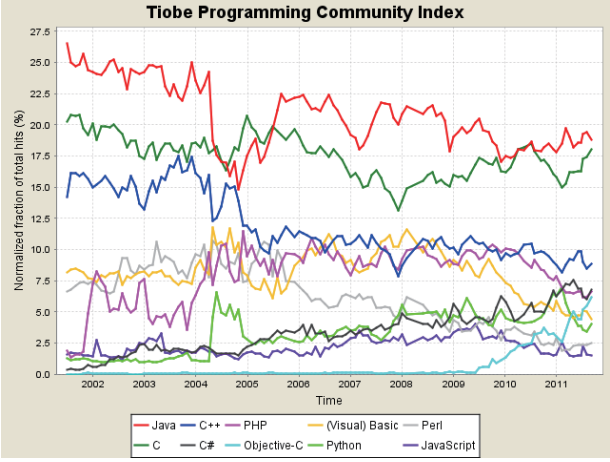
NXT-G 编程环境

如果你是喜欢自己动手又对自动控制感兴趣的程序开发者，不妨尝试一下乐高 NXT 积木和 NXT-G 编程环境，LEGO 一定会为你带来丰富的体验和乐趣。点击下面查看详细内容 >>

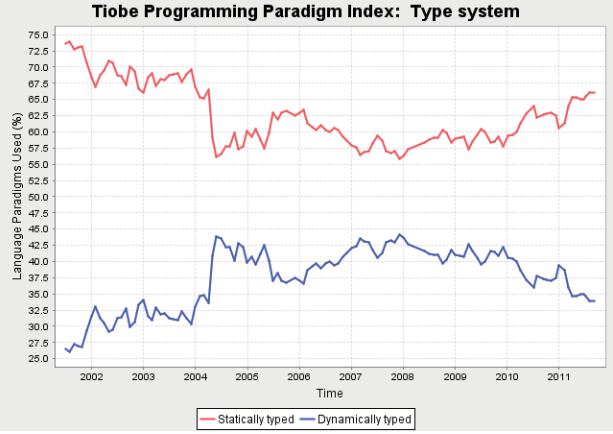
http://developer.51cto.com/art/201109/290536.htm

下面是本期编程语言排行榜的其他排名数据和趋势走向。■

前 10 名编程语言走势图



下面给出了编程语言类别的一年变化趋势



2011年暑期关于Java的那些事

2011 年的暑假，注定是 Java 的暑假。7 月底 Java 7 隆重登场，各位 Java 人终于盼来了 Java 有一个里程碑式的版本。当然在很多技术达人看来，要稳定的使用 Java 7 还需要等待它多更新几个版本。有淘宝的 Java 项目负责人表示，一个大型企业选择所用的新技术，就是对企业未来发展做出的选择，所以需要慎重。

在 Java 7 发布过了一月左右，Java 之父高斯林一纸博文《我又跑路了》，宣告自己正式从 Google 离职。高斯林在离开 Oracle 后加入 Google，当初离开甲骨文也是跟甲骨文 CEO 拉里不和。高斯林曾公开宣称，拉里是“魔鬼拉里”。

现在高斯林加入了一家科技公司，51CTO 衷心祝愿高司令在新的工作岗位上顺心如意。

在 9 月初 51CTO 还组织了第一次 Java 技术沙龙，邀请了 J-Hi 开源平台架构师张昊老师来给大家分享 Java 快速开发经验。

总而言之，这个暑期是 Java 的暑期。51CTO《开发月刊》特此以 2011 年暑期 Java 发生的事情做一总结。

51CTO 开发频道敬上

Java之父高斯林离开谷歌

我又跑路了，这是高司令今天新博文的标题。继之前离开Oracle后，Java之父今天在自己的博客里宣布已经从Google离职。高司令离开的原因是自己发现了新的方向，Liquid Robotics。

Java之父高斯林于当地时间8月30日在自己的博客发表了一篇名为《我又跑路了》的博文。在博文里高斯林宣布自己将离开Google，并解释称自己已经找到了更感兴趣的方向。

Java之父——詹姆斯·高斯林出生于加拿大，是一位计算机编程天才。在卡内基·梅隆大学攻读计算机博士学位时，他编写了多处理器版本的Unix操作系统。1991年，在Sun公司工作期间，高斯林和一群技术人员创建了一个名为Oak的项目，旨在开发运行于虚拟机的编程语言，同时允许程序在电视机机顶盒等多平台上运行。后来，这项工作就演变为Java。随着互联网的普及，尤其是网景开发的网页浏览器的面世，Java成为全球最流行的开发语言。因此被人称作Java之父。高斯林个人博客地址：

http://nighthacks.com/roller/jag/

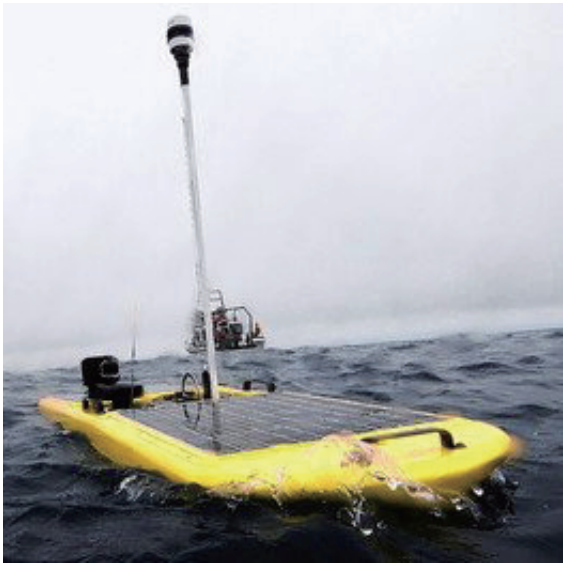
高斯林已经宣布加盟一家叫Liquid Robotics的公司，在该公司的官网上（http://liquidr.com/）也已经放出了高司令加盟的消息。高斯林自己解释说，是看中了这家公司在未来的成长性。

作为未来Liquid Robotics的首席软件架构师，高斯林将致力于板载传感器软件开发和自主导航设计，数据中心海量数据处理。该公司现有系统已经开发完毕，但高斯林认为还存在各种各样的漏洞，这是一件很有趣的事情。

此前在2010年4月，高斯林曾宣布离开Oracle公司，并加盟Google。谷歌此举被认为是想通过Java之父，规避Android平台上与Oracle的专利争端。

延伸阅读

美国Liquid Robotics公司



Liquid Robotics公司开发的新型海上船只

该公司制造了一种名为“Wave Glider”的新型海上船只，它能够利用海水的上下浮动力来驱动船只前进。

此外，操作者可使用互联网连接设备，并通过卫星对这种船只进行远程操作。

今年夏季，英国石油公司(BP)就部署了两艘Wave Glider船，用以监测该公司墨西哥湾漏油的情况。■

Java 8采用跟C#类似的Lambda句法

鉴于Java 7 SE（标准版）现已正式发布，甲骨文和Java社区进程组织（JCP）的成员们已开始仔细考虑为这种编程语言的下一个版本Java SE 8添加什么功能特性。

前段时间曝光将为Java 8提上议程的工作是：设计面向云计算的Java。近日又有提出了对Java 8语法上的改变：Java 8将采用跟C#一样的Lambda句法。

lambda-dev在邮件列表宣布，Java的Lambda语法决定基于C#语法。这种语法在C#语言里已经使用很长时间了（C#在1.0时期便引入了委托类型，并在2.0里支持匿名函数，3.0对Lambda表达式提供支持）。

这种语法同时支持表达式和代码块。表示式的形式不需要大括号，并且在执行后返回结果。语法块的形式需要使用大括号，除非使用了return关键字，否则不会直接返回结果。邮件里也给出了示例：

C#的语法是：

```
lambda = ArgList Arrow Body
ArgList = Identifier
| "(" Identifier [ "," Identifier ]* ")"
| "(" Type Identifier [ "," Type Identifier ]* ")"
Body = Expression
| "{" [ Statement ";" ]+ "}"
```

这里是使用lambda表达式这种语法的一些例子：

```
x => x + 1
(x) => x + 1
(int x) => x + 1
(int x, int y) => x + y    (x, y) => x + y
(x, y) => { System.out.printf("%d + %d = %d%n", x, y, x+y); }
() => { System.out.println("I am a Runnable"); }
```

Lambda表达式的关键优势，在于它们会对参数进行类型推断。不过在某些情况下，编译器依然无法推断出正确类型（尤其是出现操作符重载的时候，例如：(x, y) => x + y在Java编译器不知道x和y是int还是double时候）。

一般来说，类型推断引擎都能自动得出正确的代码，如果需要更多提示，程序员也始终可以显示地增加类型信息。

不久的将来会发布支持新语法的编译器以供体验。■

C# 中 Lambda 参数类型

C# 中 Lambda 表达式参数类型可以是隐式类型或显式类型。在显式列表中，每个参数的类型是显式指定的，在隐式列表中，参数的类型由Lambda表达式出现的语境自动推断类型。

Lambda表达式的参数列表可以有一个或多个参数，或者无参数。在有单一的隐型参数的lambda表达式中，圆括号可以从参数列表中省略。

C#的Lambda表达式L可以被转换为委托类型D，需以下条件：L的参数类型要与D的参数个数相等，类型相同，返回类型相同，无论是表达式，还是语句块。注意隐式类型要参与类型辨析。

简介

51CTO 八月 Java 备忘录将记录本月有关 Java 的点点滴滴，而 2011 年 8 月 Java 发生的大事自然是高斯林从 Google 离职。

8月Java备忘录：高斯林离职Google

作者 / 彭凡

2011 年 8 月 30 日，Java 之父高斯林同志正式宣布调任 Liquid Robotics 公司首席软件架构师一职。在 Google 欢送大会上，高斯林对 Google 一年来的支持表示感谢并祝福 Google 越办越好。

高斯林同志是 2010 年 4 月从 Oracle 离职后，加入 Google 的。在高斯林离开甲骨文前，爆出了其与 CEO 不和的消息，并称 Lerry 为“黑暗王子 Larry”。这样的诅咒看来是平时过度积累的结果。高斯林离职的主要原因还是在于 Java 未来发展的方向上存在争论，并认为 Oracle 过于关注盈利。

高斯林 2011 年 8 月离开 Google 则是因为他找到了新的方向，为 Liquid Robotics 加强板载传感器软件开发和自主导航设计，数据中心海量数据处理。让我们祝愿这位技术牛人工作顺利。

Scala，被遗忘的 Android 编程语言

Scala，这款运行于 Java 虚拟机之上的静态类型语言，正逐渐成为谷歌 Android 应用程序开发领域的新选择。

对于移动设备上的应用程序，Scala 则比 JRuby 及 Groovy 等其它 JVM 语言更具优势，因为静态类型语言运行速度更快、内存占用更少、优化程度也往往好于平均水平，Burns 说道。他还强调说 Android 系统往往运行于处理器速度缓慢且内存较小的嵌入式设备上，如此一来降低配置要求就显得尤为重要。更多内容，请查看《Scala：未受重视却潜力巨大的 Android 编程语言》

电影《Java 风云》预告片

一个 Windows 老爸，每天给儿子读 .NET 睡前故事。可是这小子最后还是跟 Java 跑了，还找了个 Java 女朋友。家庭正在分裂，亲情正在淡漠，《Java 风云》你可以看看。（51CTO 编辑友情提示：有部分少儿不宜的内容），预告片观看地址：

http://developer.51cto.com/art/201108/284900.htm

Java 8 呼之欲出，图谋云计算

Java 7 已经在 2011 年 7 月 28 日正式发布了。大家还没有完全搞明白 Java 7 是蜜糖还是毒药的时候，已经有人开始盘算 Java 8 了。红帽公司的一名工程师认为，Java 需要模块功能和多租户功能，才能适用于云环境。那么未来要发布的 Java 8 似乎也在图谋云计算了。

从许多方面来看，Java 8 将真正检验甲骨文管理一个复杂的开源项目的水平如何，这是许多代码贡献者的利益彼此冲突的一个项目。阅读全文，请查看《Java 8 整装待发 图谋云计算》

JRuby 和 Java 7 我们可以期待什么

Java 7 已经粉墨登场了，这次带来的新功能感觉有些普普通通，当然还是有几个重要改进。那么对于在 Java 7 上运行的 JRuby，我们能够期待些什么呢？欲查看本文更多内容，请访问：

http://developer.51cto.com/art/201108/288222.htm ■

51CTO技术沙龙 第十三期

Java快速开发之路

在 2011 年 9 月 3 日下午 51CTO 举办的第 13 期 51CTO 技术沙龙中，有幸请到 Java 开源平台——J-Hi 的两位创始人张昊老师和肖金华老师。二位讲师分别从理论和实战来为大家介绍 J-Hi，打开 Java 快速开发之路。

本期沙龙资料下载：

主题：《Java 快速开发的分析与探索》（张昊）

http://down.51cto.com/data/245952

主题：《Java 快速开发平台使用》（肖金华）

51CTO 技术沙龙第十三期：Java 快速开发之路视频专题

http://developer.51cto.com/developer/51cto-salon-13/

本期沙龙的第一位嘉宾张昊的演讲题目是《Java 快速开发的分析与探索》，在本期沙龙报名推出时，引发了很多网友疑惑 “Java 快速开发与敏捷开发有什么区别呢？”，张昊老师在沙龙中为大家解答了这个问题：

敏捷开发是一种以人为核心、迭代、循序渐进的开发方法。而 J-Hi 则是以工具为核心，利用组件化、标准化来提高开发效率，节省开发时间，降低成本，二者可以相互结合。

敏捷开发中的短周期迭代和需求测试似乎是解决之道，但不一定适合每一个公司，而 J-Hi 可应用到每个公司，甚至适用于每个 Java 开发者。

J-Hi 快速开发平台团队核心成员 张昊

软迅博技术发展有限公司首席架构师，开源 J-Hi 快速开发平台团队核心成员，平台项目的主要发起者与组织者。十多年来一直从事于 J2EE 企业级应用开发及项目管理工作。

软迅博技术发展有限公司项目总监 肖金华

开源 J-Hi 快速开发平台团队核心成员，十一年软件开发经验，带领完成过多个大型项目。熟悉保险及电力行业，曾任某 IT 公司保险业务线架构师。

ASP.NET核心对象

今天我们将谈到的是 ASP.NET 核心对象，搞清楚这些对大家更高效的开发是有帮助的。

想当初在只使用 WebForms 框架并以服务端为中心的开发模式时，发现 ASP.NET 好复杂。一大堆服务端控件，各有各的使用方法，有些控件的事件也很重要，必须在合适地时机去响应，还真有些复杂。后来逐渐发现这些复杂的根源其实就是服务器控件相关的抽象逻辑。

随着 Ajax 越用越多，可能有些人也做过这些事情：【新建一个 ashx 文件，读取一些用户的输入数据，Form，QueryString，然后调用业务逻辑代码，将处理后的结果序列化成 JSON 字符串再发给客户端】，这样也能完成一次请求。不知大家有没有做过这类事情，反正我是做过的。慢慢地，我也嫌烦了，这些事情中除了调用业务逻辑部分，都是些体力活嘛。于是想，写点代码把这些事情交给它们去做吧，我只处理与请求有关的数据处理就好了。终于，我写了个简陋的框架，并自称为【我的 Ajax 服务端框架】以及【我的 MVC 框架】。写完这些东西后，发现 ASP.NET 的东西变少了，但是仍可以实现很多功能。

其实，我们可以从另一角度来看 ASP.NET，它就是一个底层框架平台，它负责接收 HTTP 请求（从 IIS 传入），将请求分配给一个线程，再把请求放到它的处理管道中，由一些其它的【管道事件订阅者】来处理它们，最后将处理结果返回给客户端。而 WebForms 或者 MVC 框架，都属于 ASP.NET 平台上的【管道事件订阅者】而已，Web Service 也是哦。如果你不想受限于 WebForms 或者 MVC 框架，或者您还想用 ASP.NET 做点其它的事情，比如：自己的服务框架，就像 WebService 那样。但希望用其它更简单的序列化方式来减少网络流量，或者还有加密要求。那么了解 ASP.NET 提供了哪些功能就很有必要了。

本文将站在 ASP.NET 平台的角度，来看看 ASP.NET 的一些基础功能。虽然不会涉及任何其它上层框架，但所讲述的内容其实是适合其它上层框架的。

前面我说到：ASP.NET 负责接收请求，并将请求分配给一个线程来执行。最终执行什么呢？当然就是我们的处理逻辑。但我们在处理时，用户输入的数据又是从哪里来的呢？只能是 HTTP 请求。但它又可分为二个部分：请求头和请求体。在 ASP.NET 中，我们并不需要去分析请求头和请求体，比如：我们可以直接访问 QueryString，Form 就可以得到用户传过来的数据了，然而 QueryString 其实是放在请求头上，在请求头上的还有 Cookie，Form 以及 PostFile 则放在请求体中。如果对这些内容不清楚的可以参考我的博客：【细说 Cookie】和【细说 Form（表单）】。您应该可以看出：要是让您从请求头请求体中读取这些

数据，还是很麻烦的。幸好，ASP.NET 做为底层平台，在每次处理请求时，都将这些数据转成方便我们处理的对象了。今天我将只谈这些基础对象以及它们可以实现的功能。

在我的眼里，ASP.NET 有三大核心对象：HttpContext, HttpRequest, HttpResponse。

除此之外，还有二个对象虽然称不上核心，但仍然比较重要：HttpRuntime，HttpServerUtility

事实上，这些类的实例在其它的一些类型中也经常被引用到，从出现的频率也可以看出它们的重要性。

中国人喜欢把较重要的东西放在最后，做为压轴出场。今天我也将按照这个风俗习惯做为这些对象的出场顺序来分别说说它们有哪些【重要的功能】。

HttpRuntime

第一个出场的是 HttpRuntime，其实这个对象算是整个 ASP.NET 平台最核心的对象，从名字可以看出它的份量。但它包含的很多方法都不是 public 类型的，它在整个请求的处理过程中，做了许多默默无闻但非常重要的工作。反而公开的东西并不多，因此需要我们掌握的东西也较少。不能让它做为压轴出场就让它第一个出场吧。这就是我的想法。

HttpRuntime 公开了一个方法静态 UnloadAppDomain()，这个方法可以让我们用代码重新启动网站。通常用于用户通过程序界面修改了一个比较重要的参数，这时需要重启程序了。

HttpRuntime 还公开了一个大家都熟知的静态属性 Cache。可能有些人认为他 / 她在使用 Page.Cache 或者 HttpContext.Cache，事实上后二个属性都是 HttpRuntime.Cache 的【快捷方式】。HttpRuntime.Cache 是个非常强大的东西，主要用于缓存一些数据对象，提高程序性能。虽然缓存实现方式比较多，一个 static 变量也算是能起到缓存的作用，但 HttpRuntime.Cache 的功能绝不仅限于一个简单的缓存集合，如果说实现“缓存项的滑动过期和绝对过期”算是小儿科的话，缓存依赖的功能应该可以算是个强大的特性吧。更有意义的是：它缓存的内容还可以在操作系统内存不足时能将一些缓存项释放（可指定优先级），从而获得那些对象的内存，并能在移除这些缓项时能通知您的代码。可能有人认为当内存不足时自动释放一些缓存对象容易啊，使用 WeakReference 类来包装一下就可以了。但 WeakReference 不提供移除时的通知功能。

HttpRuntime.Cache 还有个非常酷的功能是：它并非只能在 ASP.NET 环境中使用，也能在其它编程模型中使用，比如大家熟知的 WinForm 编程模型。如何使用呢，直接访问 HttpRuntime.Cache 这个静态属性肯定是不行的。我们只要在程序初始化时创建一个 HttpRuntime 的实例，当然还要保证它不会被 GC 回收掉。然后就可以像在 ASP.NET 中一样使用 HttpRuntime.Cache 了，就这么简单。是的，就是这样简单，您就可以在其它编程模型中使用 Cache 的强大功能：线程安全的集合，2 种过期时间的选择，缓存依赖，内存不足时自动释放且有回调通知。

这里我还想说说缓存依赖。我曾经见过一个使用场景：有人从一堆文件（分为若干类别）中

ASP.NET 核心对象 III

加载数据到 Cache 中，但是他为了想在这些数据文件修改时能重新加载，而采用创建线程并轮询文件的最后修改时间的方式来实现，总共开了 60 多个线程，那些线程每隔 15 去检查各自所“管辖”的文件是否已修改。如果您也是这样处理的，我今天就告诉您：真的没必要这么复杂，您只要在添加缓存项时创建一个 CacheDependency 的实例并调用相应的重载方法就可以了。具体 CacheDependency 有哪些参数，您还是参考一下 MSDN 吧。这里我只告诉您：它能在一个文件或者目录，或者多个文件在修改时，自动通知 Cache 将缓存项清除，而且还可以设置到依赖其它的缓存项，甚至将这些依赖关系组合使用，非常强大。

可能还有人会担心往 Cache 里放入太多的东西会不会影响性能，因此有人还想到控制缓存数量的办法。我只想说：缓存容器决定一个对象的保存位置是使用 Hash 算法的，并不会因为缓存项变多而影响性能，更有趣的是 ASP.NET 的 Cache 的容器还并非只有一个，它能随着 CPU 的数量而调整，看这个架式，应该在设计 Cache 时还想到了高并发访问的性能问题。如果这时你还在统计缓存数量并手工释放某些缓存项，我只能说您在写损害性能的代码。

HttpServerUtility , HttpUtility

不要觉得奇怪，这次我一下子请了二个对象出场了。由于 HttpServerUtility 的实例通常以 Server 的属性公开，但它的提供一些 Encode, Decode 方法其实调用的是 HttpUtility 类的静态方法。所以我就把它们俩一起请出来了。

HttpUtility 公开了一些静态方法，如：

HtmlEncode()，应该是使用频率比较高的方法，用于防止注入攻击，它负责安全地生成一段 HTML 代码。

有时我们还需要生成一个 URL，那么 UrlEncode() 方法就能派上用场了，因为 URL 中并不能包含所有字符，所以要做相应的编码。

HttpUtility 还有一个方法 HtmlAttributeEncode()，它也是用于防止注入攻击，安全地输出一个 HTML 属性。

在 .NET4 中，HttpUtility 还提供了另一个方法：JavaScriptStringEncode()，也是为了防止注入攻击，安全地在服务端输出一段 JS 代码。

HttpUtility 还公开了一些静态方法，如：

HtmlDecode(), UrlDecode()，通常来说，我们并不需要使用它们。尤其是 UrlDecode，除非您要自己的框架，一般来说，在我们访问 QueryString, Form 时，已经做过 UrlDecode 了，您就不用再去调用了。

HttpServerUtility 除了公开了比较常用的 Encode, Decode 方法外，还公开了一个非常有用的方法：Execute()，是的，它非常有用，尤其是您需要在服务端获取一个页面或者用户控件的 HTML 输出时。

HttpRequest

现在总算轮到第一个核心对象出场了。MSDN 给它作了一个简短的解释：“使 ASP.NET 能读取客户端在 Web 请求期间发送 HTTP 值。”

本文未完，更多内容请访问原文，链接：

http://developer.51cto.com/art/201108/285626.htm

本文转载自博客园作者 Fish Li 相关博文。■

编者按

解释语言和虚拟机都很不错，但一种新版本的 C++ 却表明人们对老式的原生二进制代码重新产生了兴趣。这就是 C++ 11。

原生代码卷土重来 C++欲东山再起

编程语言往往是各领风骚三五年。最初，炙手可热的新语言是 Java；后来换成了 Python，随后 Ruby 抢走了风头，之后又换成了 JavaScript。而最近备受宠爱的语言可能大家最想不到的。信不信由你，2011 年很可能是 C++ 大行其道年。

最新版的 ISO C++ 标准被全体一致批准，这是 C++ 语言 13 年来第一个重大修订版。新标准现在的官方名称是 C++11，它引入的一些功能特性旨在更容易地针对现代并行处理架构开发软件，包括面向并行计算的 Lambda 表达式和新的数据类型。

倒不是说 C++ 真的消亡了。与年代更久的同类语言 C 相比，C++ 对于系统编程和需要性能密集型原生代码的应用程序（如 3D 游戏引擎）来说仍是最流行的语言之一。

然而在其专门的小众领域之外，传统的系统编程在近些年已渐渐失宠。现在的程序员日益远离原生代码编译，改而青睐 Java 和 .NET 等受控代码环境。这种环境让他们可以不用操心内存管理和输入验证方面这样单调乏味的工作。另一些程序员为了获得 Python、Ruby 和 JavaScript 等动态语言在语法上的便利，愿意牺牲一些性能。

但是 C++ 11 出现在颇有意思的时间点。越来越多的人觉得，编程语言这个钟摆偏离原生代码也许太远了。现在该是钟摆往另一个方向摆回来的时候了。因而，C++ 发现自己多了几个原本最不可能的盟友。

原生代码最不可能的拥趸包括谷歌

谷歌恐怕是你最不会想到对原生代码有兴趣的一家公司。多年来，谷歌一直鼓吹桌面软件概念已过时了这一理念。在谷歌憧憬的理想环境下，应用程序完全在浏览器里面运行；为此，它开发出了 Chrome OS 来证明这一点。

不过连谷歌都认识到，有时候经过解释的 JavaScript 还不够。我在以前的文章中介绍过谷歌原生客户端（NaCl），这种沙箱环境让 Chrome 浏览器可以下载和执行原生二进制代码，以便处理性能密集型操作。这绝非谷歌在闲暇之余的试验活动；上周，谷歌在 Chrome 的最新测试版中发布了新版本的 NaCl；这项技术头一次在默认情况下被启用。还有传闻称，谷歌悄然添加到最新版 Chrome OS 中的 Netflix 媒体流支持功能也依赖 NaCl 应用编程接口（API）。

NaCl 也不是表明谷歌对原生代码的唯一认可。这家搜索巨头的 Go 编程语言之前就被广泛称为是“类似 Java”，但这只说对了一部分。从语法上来讲，Go 某些方面的确类似 Java，但 Go 代码并不在虚拟机里面运行，而是直接编译成了

原生代码。此外，谷歌甚至确保 Go 二进制代码不但可以在桌面上运行，还可以在谷歌应用程序引擎（Google App Engine）云计算环境里面运行。

很奇怪的是，这些举措使得谷歌与位于雷德蒙的竞争对手关系融洽和睦。作为世界上最大的桌面软件开发商，微软对待 C++ 的态度一直比许多公司更友好。可是近些年来，原生 C/C++ 开发人员觉得自己受到了一点冷落，因为微软把大部分精力投入到了 C#，这是面向 .NET 平台的受控代码 C 衍生语言。

但这种情形似乎在发生变化。Visual Studio 2010 已经支持 C++11 的大部分功能特性。今年 7 月，微软在其 Channel 9 开发人员网站上发布了一个新的视频系列，名为“Going Native”（使用原生代码），专门介绍原生代码方面的发展，特别强调 C++。与此同时，目光敏锐的观察人士注意到了微软在 9 月召开的 Build 大会前夕居然没有 .Net 方面的宣传。Build 大会之前名叫专业开发者大会（PDC）。

原生代码永不没落

早在 Java 时代之前就懂得编程的那些人自然完全明白为何人们重新对原生代码产生了兴趣。传统的软件开发方法有着悠久的传奇历史；有时候，经过充分优化的原生二进制代码仍是有可能从处理器获取最高性能的最佳方法。

话虽如此，原生代码也有其缺点。首当其冲的是我之前提到的安全问题。由于与 C# 和 Java 等受控语言相比，C 和 C++ 等语言让开发人员更接近底层硬件，所以要认真考虑的问题就多得多；缺乏经验的编程员可能会造成严重破坏。

移植性是另一个问题。对于处在 Wintel 环境下的 Windows 开发人员来说，编写可以在多种处理器架构上编译的代码从来不是个大问题，甚至在过去也是如此。然而换成了 Unix 环境，情况就不一样了。

现在，谷歌的 NaCl 环境重新带来了同样一些问题。Web 开发人员习惯于自己开发的应用程序在任何基本上符合标准的浏览器中运行，但是 NaCl 模块与特定的处理器架构死死地绑在一起。想让同一个模块在 x86、x64 和 ARM 等处理器上都可以运行，你就得编译这三个不同的版本，然后把这三个版本都放到 Web 服务器上。对于长期习惯使用 PHP 和 Perl 的开发人员来说，这就需要一番适应。

但要牢记的最重要的问题是，为手头的任务选择合适的工具。谁也不想回到以前的糟糕年代：面对使用由 C 编写的 CGI 脚本的 Web，为文本数据而纠结。另一方面，不管手头的任务是什么，把每一个应用程序硬塞进同样的解释语言或受控代码环境同样不是正确的方法。

现在的计算机异常尖端先进、用途异常广泛、性能异常强大。要充分利用它们的功能，开发人员就应该使用尽可能广泛的工具。原生代码的卷土重来往正确方向迈出了可喜的一步。■

详解C++11中值得关注的几大变化

作者 / 陈皓

原文章来自前 C++ 标准委员会的 Danny Kalev 的 The Biggest Changes in C++11 (and Why You Should Care)，赖勇浩做了一个中文翻译在这里。所以，我就不翻译了，我在这里仅对文中提到的这些变化“追问为什么要引入这些变化”的一个探讨，只有知道为了什么，用在什么地方，我们才能真正学到这个知识。而以此你可以更深入地了解这些变化。所以，本文不是翻译。

Lambda 表达式

Lambda 表达式来源于函数式编程，说白就了就是在使用的地方定义函数，有的语言叫“闭包”。表达式的简单语法如下，我在这里就不再讲这个事了。

```
[capture](parameters)->return_type {body}
```

原文的作者给出了下面的例子：

```
int main()
{
   char s[]="Hello World!";
   int Uppercase = 0; //modified by the lambda
   for_each(s, s+sizeof(s), [&Uppercase] (char c)
{
    if (isupper(c))
     Uppercase++;
    });
   cout << Uppercase << " uppercase letters in:
" << s <<endl;  }
```

在传统的 STL 中 for_each() 这个玩意最后那个参数需要一个“函数对象”，所谓函数对象，其实是一个 class，这个 class 重载了 operator()，于是这个对象可以像函数的 式样的使用。实现一个函数对象并不容易，需要使用 template，比如下面这个例子就是函数对象的简单例子（实际的实现远比这个复杂）：

```
template <class T>
class less
{
public:
   bool operator()(const T&l, const T&r)const
   {
      return l < r;
   } };
```

所以，C++ 引入 Lambda 的最主要原因就是 1) 可以定义匿名函数，2) 编译器会把其转成函数对象。相信你会和我一样，会疑问为什么以前 STL 中的 ptr_fun() 这个函数对象不能用？(ptr_fun() 就是把一个自然函数转成函数对象的）。原因是，ptr_fun() 的局限是其接收的自然函数只能有 1 或 2 个参数。

那么，除了方便外，为什么一定要使用 Lambda 呢？它比传统的函数或是函数对象有什么好处 呢？我个人所理解的是，这种函数之年以叫“闭包”，就是因为其限制了别人的访问，更私有。也可以认为他是一次性的方法。Lambda 表达式应该是简洁的， 极私有的，为了更易的代码和更方便的编程。

自动类型推导 auto

在这一节中，原文主要介绍了两个关键字 auto 和 deltype，示例如下：

```
auto x=0; //x has type int because 0 is int
auto c='a'; //char
auto d=0.5; //double
auto national_debt=14400000000000LL;//long
long
```

auto 最大的好处就是让代码简洁，尤其是那些模板类的声明，比如：STL 中的容器的迭代子类型。

```
vector<int>::const_iterator ci = vi.begin();
```

详解 C++11 中值得关注的几大变化 II

可以变成：

```
auto ci = vi.begin();
```

模板这个特性让 C++ 的代码变得很难读，不信你可以看看 STL 的源码，那是一个乱啊。使用 auto 必需一个初始化值，编译器可以通过这个初始化值推 导出类型。因为 auto 是来简化模板类引入的代码难读的问题，如上面的示例，iteration 这种类型就最适合用 auto 的，但是，我们不应该把其滥 用。

比如下面的代码的可读性就降低了。因为，我不知道 ProcessData 返回什么 ?int? bool? 还是别的什么 ? 这让你后面的程序不知道怎么做。

```
auto obj = ProcessData(someVariables);
```

但是下面的程序就没有问题，因为 pObject 的型别在后面的 new 中有了。

```
auto pObject = new SomeType<OtherType>::SomeOtherType();
```

自动化推导 decltype

关于 decltype 是一个操作符，其可以评估括号内表达式的类型，其规则如下：

如果表达式 e 是一个变量，那么就是这个变量的类型。

如果表达式 e 是一个函数，那么就是这个函数返回值的类型。

如果不符合 1 和 2，如果 e 是左值，类型为 T，那么 decltype(e) 是 T&; 如果是右值，则是 T。

原文给出的示例如下，我们可以看到，这个让我的定义省了很多事。

```
const vector<int> vi;
typedef decltype (vi.begin()) CIT;
CIT another_const_iterator;
```

还有更适用的用法是用来 typedef 函数指针，这个会省很多事。比如：

```
decltype(&myfunc) pfunc = 0;
typedef decltype(&A::func1) type;
```

auto 和 decltype 的差别和关系

Wikipedia 上是这么说的 (关于 decltype 的规则见上)

```
#include <vector>
 int main()
{
   const std::vector<int> v(1);
   auto a = v[0];       // a 的类型是 int
    decltype(v[0]) b = 1; // b 的类型是 const int&, 因为函数的返回类型是
   // std::vector<int>::operator[](size_type) const
   auto c = 0;          // c 的类型是 int
   auto d = c;          // d 的类型是 int
   decltype(c) e;       // e 的类型是 int, 因为 c 的类型是 int
    decltype((c)) f = c;  // f 的类型是 int&, 因为 (c) 是左值
   decltype(0) g;       // g 的类型是 int, 因为 0 是右值 }
```

如果 auto 和 decltype 在一起使用会是什么样子 ? 能看下面的示例，下面这个示例也是引入 decltype 的一个原因——让 C++ 有能力写一个 “forwarding function 模板”，

```
template< typename LHS, typename RHS>
  auto AddingFunc(const LHS &lhs, const RHS &rhs) -> decltype(lhs+rhs)
{return lhs + rhs;}
```

这个函数模板看起来相当费解，其用到了 auto 和 decltype 来扩展了已有的模板技术的不足。怎么个不足呢 ? 在上例中，我不知道 AddingFunc 会接收什么样类型的对象，这两个对象的 + 操作符返回的类型也不知道，老的模板函数无法定义 AddingFunc 返回值和这两个对象相加后的返回值匹配，所以，你可以使用上述的这种定义。(未完，查看详细 http://developer.51cto.com/art/201108/284929.htm)

标准的日本软件开发流程

作者 /heartstill

日本的软件项目开发进度控制非常严格，项目很少出现延期，一旦延期，伴随而来的就是大宗的罚款，因此，日本的软件项目非常重视按期交付。在日本软件项目进度控制中起关键作用的就是软件的阶段定义。

日本软件项目阶段分项目提案、要件定义、概要设计、详细设计、编写代码、单体测试、结合测试、系统测试、编写手顺等。

项目提案指项目可行性分析、项目立项，是用户需求的正式提出阶段，本阶段出具《项目提案书》。

要件定义指业务需求的详细确定和系统需求的详细确定，系统需求主要包括软件安全性，运行速度，网络环境，运行环境，平台，架构等方面的要求，以及技术选择的调查，本阶段出具《业务要件定义书)) 和《系统要件定义书》。

概要设计指功能设计，系统架构设计，界面设计和数据库设计，其中界面设计和数据库设计涉及内容最多，要求最详细，本阶段出具《概要设计定义书》、((数据库设计定义书》和《界面设计定义书》。

详细设计主要指编码前的类设计，类中方法属性设计，类之间调用关系设计，本阶段出具《详细设计定义书》。

编写代码指各模块负责人编写相关代码，在编码之前还要编写单体测试式样书，本阶段出具程序源码和 (《单体测试式样书》。

单体测试指各模块编码人员完成各自模块的单体测试工作，单体测试完成要求各模块独立运行时缺陷均消除，本阶段出具《单体测试票》。

结合测试指各模块单体测试完成后，各模块同时运行时，模块之间的运行状况的测试，包括业务流，负载，运行速度，稳定性，一致性等内容，本阶段出具《结合测试票》。系统测试指系统各模块统一运行缺陷均消除后，

模拟用户环境运行的测试过程，本阶段要尽量模拟用户实际平台，用户数量，硬件环境，软件环境，网络状况，用户数据进行系统测试，本阶段出具《系统测试票》。

编写手顺指编写用户手册，本阶段出具《安装手顺》、《使用手顺》和《维护手顺》。

对日开发的基本流程中包括了以上 11 个阶段，每个阶段为一个里程碑，每个里程碑在安排计划时都规定了明确的完成期限，这些阶段性的里程碑是项目进度的关键点。每个阶段完成后必须进行阶段的 Review，这种阶段 Review 起到了阶段验收和总结的作用。阶段 Review 是日本项目阶段控制的核心。只采用阶段 Review 的方式进行验收也有其不足之处，所有验收工作都放在阶段完成再进行，阶段中的错误后续持续放大无法得到控制。而且通常情况下，阶段 Review 时问题会比较多，Review 后修改. 时间比较长，修改次数也较多，造成很大程度的反复工作。再有，标准对日软件开发过程中，阶段内任务的安排和验收比较；无序，很多问题会被有意推迟到 Review 时解决。■

揭秘Facebook是如何开发软件的

本文是从 How Facebook Ships Code 这篇文章翻译而来。本文是作者从很多在 Facebook 工作的朋友那里搜集到的关于这个公司如何开发和发布软件的只言片语。

Facebook 的工作方式让我着迷。那是一个非常独特的工作氛围，无法复制（也并不适用于其它公司）。下面是我从很多在 Facebook 工作的朋友那里搜集到的关于这个公司如何开发和发布软件的只言片语。

看起来对 Facebook 感兴趣的大有人在。这个公司以程序员为主导的企业文化受到人们的极大关注，很多公司都在努力实现这样的企业文化。

尽管 Facebook 对于其内部的开发过程讳莫如深，但他们的技术团队还是会对其新功能和一些内部系统做一些公开的说明，可这些说明通常是关于“是什么”之类的文章，而不是关于“如何做”的…

所以，作为一个外人，你很难知道 Facebook 是如何做到比其他公司更有效的对其产品进行改进和优化。我作为一个外部人士，尝试着去了解更多的关于 Facebook 内部是如何运转的信息，我把这几个月的观察收获进行了汇编。

出于对于信息来源者的隐私保护，我删除了所有涉及到的人名和特定产品特征 / 产品名称。而且我把这篇文章延迟了 6 个多月才对外发布，所以，文章中所涉及的内容都不会太新太敏感。

我希望这篇文章能给那些试图看清 Facebook 如何做到决策权“下放”而不引起管理混乱的人增加一些亮光。你很难评论 Facebook 这种做法的好坏，以及 Facebook 的产品质量跟这种做法的关系。我想、也希望如此多的互联网消费型公司都能从 Facebook 公司的例子中学到有用的知识。

非常感谢那些在 Facebook 内部工作、帮助我得到这些信息的人，同时也感谢像 epriest 和 fryfrog 这样对本文进行校正和修改的人。

语录：

◆ 截止到 2010 年六月，这个公司的员工已经接近 2000 名，而在此 10 个月之前只有大概 1100 名。一年内几乎翻了一番！

◆ 公司最大的两群人是技术开发人员和实施人员 (Ops)，各自有 400 ~ 500 人。这两部分人占去了公司构成的 50%。

◆ 产品经理跟技术人员的比例大概是 1:7 到 1:10。

◆ 所有的技术人员都要通过 4 到 6 周的“新兵训练营”培训，培训中他们通过修改 bug 来了解 Facebook 系统，听资深 / 终身司职技术人员做演讲。每次训练营培训大概会有 10% 的学员不能通过考核，会被淘汰出公司。

◆ 新兵训练营后，所有的技术人员都要接触真实现场数据库（先会有个专门的讲座，关于“责任越大，能力越大”，还有一个明确的“违反即开除”的清单，例如泄漏私人信息）。

[感谢 fryfrog 的修改]” 公司有很多非常有效的防护措施来防止内部拥有这种能力的人做出各种恐怖的事情”，如果你不幸成为需要做这种危险操作的人，你需要登记原因，而且会被密切的

揭秘 Facebook 是如何开发软件的 II

审查。一点疏忽都不能有，否则你完了。

◆ 任何技术人员都可以修改 Facebook 代码库里的任何一段代码，并按自己的意愿提交回代码库里。

◆ 非常强势的技术人员为主导的文化。“产品经理在这里基本上没有什么用处。” —— 引自一位开发人员的话。程序员人员可以在中途修改产品规格文档，重新调整要做哪个项目，随时都可以按自己的想法加入新的功能特征。

[编辑评论] 这篇博客的作者是一位产品经理，所以这段文字着实让我意外。你会在余下的语录里看到，Facebook 的企业文化对产品的管理工作是十分重视的。所以，产品管理这个角色并不是可有可无的。并且，这个公司的企业文化是让 “每一个员工” 都感到对产品有责任。

◆ 在每月的跨团队会议中，进度报告由开发人员提交。产品市场和产品管理部门会出席这些会议，但如果在会上他们说了太多的话，会后领导会收到会议反馈 “产品部门在会上说的话太多了。” 他们真的希望开发人员能公认的完全控制产品，成为公司开发的产品的主要主导成分。

◆ 每个项目的人力调配完全是根据自愿。

◆ 产品经理要游说开发人员，让他们对自己的想法感兴趣。

◆ 开发人员选择他们听起来感兴趣的任务。

◆ 开发人员会对他们的经理说：“本周我打算做这 5 块工作。”

◆ 技术经理会尽可能的由着各程序员的喜好行事，但有时会要求某项工作必须先做。

◆ 程序员自己把握所有的技术特征 —— 前端的 javascript，后端的数据库脚本，以及所有这之间的东西。如果他们需要设计人员的帮助 (只有少数几个专职设计人员)，那他需要找到一个对他们的项目感兴趣的设计师。找架构师也是如此。但通常，程序员会自己处理所有所需。

◆ 一个功能特征是否值得做，通常的判断方法是用一周快速实现，然后在抽样用户里测试它，例如找 1% 的内华达州用户进行测试。

◆ 开发人员通常喜欢关于基础架构，系统扩展性，“难题” 等的任务 —— 这些都是能产生威望的地方。你很难让一个程序员对前端项目或用户界面工作提起兴趣。这跟你在一些面向客户的业务公司里发现的现象正好相反，那些公司里所有人都喜欢干客户能接触到的东西，他们会指着某一个界面功能说：“这是我做的”。在 Facebook，后端的工作，例如新闻 feed 算法，广告定位算法，memcache 优化工作等，都是程序员们的抢手工作。

◆ 对某项具有高优先级的功能有影响的修改 (例如新闻 feed)，在代码提交合并前要经过代码审查。新闻 Feed 非常的重要，任何的改动都要经过 Zuckerberg(Facebook 创始人，总裁) 亲自审查，但也有例外的时候。

[纠正——感谢 epriest] “任何的代码的修改都必须进行强制性的代码审查 (由一个或多个技术人员执行)”。我想这篇文章中说的是 Zuck 本人并不会亲自审查每一处变动。“

所有的代码的变更都会经过至少一个人的审查，这套系统让其他人很容易的查看、审查你的代码——即使你没邀请他。想让未经审查的代码进入代码库属于一种蓄意的不良行为。”（本文未完，请点击 http://developer.51cto.com/art/201109/288557.htm ）■

8月Web技术最前沿:Edge激千层浪

作者 / 七武海

8 月炎热的夏季终于过去,让人分外的清爽,当然作为 Web 前端开发者,最高兴的莫过工具的更新和技术的进步,下面我们就向你推荐八月 Web 技术最前沿。

Adobe 发布 HTML 5 网页动画工具 Edge

Adobe 刚刚发布了一个新的工具 {Adobe Edge},允许设计师通过 HTML5、CSS 和 JavaScript 制作网页动画。无需 Flash。

Adboe Edge 的目的是帮助专业设计师制作网页动画乃至简单游戏。目前该工具的重点放在动画引擎上,但未来将增加更多 HTML5 功能,比如 Canvas、HTML5 音频 / 视频标签等。支持 Android、iOS、webOS、黑莓 PlayBook、Firefox、Chrome、Safari 和 IE9 等各个平台。

查看详细:http://developer.51cto.com/art/201108/280045.htm

下载:http://labs.adobe.com/technologies/edge/

Adobe Edge 的发布也引起了业界对 Adobe 公司战略的探讨,详情请查看:HTML5 VS Flash 之争 即见分晓?

AppMobi 发布 HTML 5 开发工具 XDK

AppMobi 推出了全新开发工具 XDK,使得开发者可以使用 HTML5 构建网络和移动平台的应用程序。最终代码既可以用来进行 HTML5 应用程序开发,就如同现在在 Chrome 网络应用程序商店里看到的那些程序一样,也可以用于多平台应用程序开发,最终提交到苹果或 Android 的应用程序商店。XDK 本身属于网络应用程序,可以在 Chrome 的网络应用程序商店免费下载。

查看详细:http://developer.51cto.com/art/201108/281491.htm

下载:https://chrome.google.com/webstore/detail/onmkoldigcfmebcinpmineoadckalllb?hc=search&hcp=main

Firebug1.8 正式版发布

著名的 Firefox 网页调试开发插件 Firebug 1.8 正式版在八月初发布了,兼容 Firefox 5.0。增加了很多的新功能。

查看新功能:http://developer.51cto.com/art/201108/280380.htm

下载:http://getfirebug.com/releases/firebug/1.8/

Grails 2.0 M1 发布

Grails 是一套用于快速 Web 应用开发的开源框架,它基于 Groovy 和 Java 编程语言,并构建于 Spring、Hibernate 和其它标准的 Java 框架之上,利用了 Java EE sphere 最好的 APIS, 从而为大家带来一套能实现超高生产力的一站式框架。Grails 2.0 M1 增加了很多新特性并修复了大量的 Bug 和提升了部分性能

查看新特性:http://developer.51cto.com/art/201108/280702.htm

下载:http://grails.org/Download

Ruby 1.9.3 第一个预览版发布

Ruby 1.9.3 第一个预览版已经发布了,这是一个参照级的版本,有可能包含 Bug,但是这些 Bug 会在下一个版本 Ruby 1.9.3-p0 中修正。

该版本自 1.9.2 以来的改进包括:

1. 许可证更改:

更改了 Ruby 的许可证,从 GPLv2 双许可证更改为 2-clause BSDL 双许可证。

2. 升级了 C API:

8 月 Web 技术最前沿 :Edge 激千层浪 II

rb_scan_args() is enhanced with support for option hash argument extraction.

ruby_vm_at_exit() added. This enables extension libs to hook a VM termination.

3. 更新了库：

包括 ARGF、Array、Bignum、Encoding、File、IO、Kernel、Module、Random、String、Time、Process 等。

4. 语言变化：

正则表达式现在支持 Unicode6.0(新的字符和脚本)

正则表达式现在支持 Age 属性 (实验阶段)

使用指令开启 / 关闭缩进警告。

发行日志：http://svn.ruby-lang.org/repos/ruby/tags/v1_9_3_preview1/NEWS

详细更新信息：http://svn.ruby-lang.org/repos/ruby/tags/v1_9_3_preview1/ChangeLog

下载地址：http://ftp.ruby-lang.org/pub/ruby/1.9/ruby-1.9.3-preview1.zip

Webware for Python 1.1 发布

Webware for Python 1.1 发布了，开发 WEB 应用程序的 Python 工具包。

Webware for Python 是一套用来开发面向对象的 WEB 应用程序的 Python 工具包。它具有良好的设计模式，包含一个快速的应用服务器、Servlets 、Python Server Pagers (PSP) 、ORM 框架、任务调度、Session 管理等等。而且具备模块化和可扩展性，支持多平台，兼容多种 Web 服务器、数据库服务器和操作系统等特点。

查看新特性：http://developer.51cto.com/art/201108/282134.htm

下　载：http://sourceforge.net/projects/webware/files/Webware/1.1/Webware-1.1.tar.gz/download

PHP 5.3.8 发布

PHP 5.3.8 版本发布，修复了 5.3.7 中的两个问题，

下载地址：

PHP 5.3.8 Final for Windows (15.00 MB): http://down.51cto.com/data/240712

PHP 5.3.8 Final for Linux (10.44 MB): http://down.51cto.com/data/240720 ■

学习 HTML 5 的 11 个顶级资源

HTML5 已经成为互联网行业的最新流行语，不少人相信这是 Web 的未来。即使没有加入该社区的人也承认，HTML5 包括了本来应该从一开始的 HTML 规范中包含的功能。在这篇文章中，我们将告诉你大约 10 个 HTML5 的资源，让您现在开始使用该技术。

1)HTML 5 工作草案标准：
http://dev.w3.org/html5/spec/Overview.html

2)HTML5 医生：http://html5doctor.com/

3)HTML5 的兼容表 :http://caniuse.com/

4)HTML 5 与 HTML 4 的差异：
http://www.w3.org/TR/2008/WD-html5-diff-20080122/

5)HTML 5 的特性一览表：
http://www.smashingmagazine.com/2009/07/06/html-5-cheat-sheet-pdf/

6) 在 Chrome 的实验：
http://www.chromeexperiments.com/

7)HTML5 和 CSS3 支持：
http://www.findmebyip.com/litmus/#target-selector

8)CSS3 和 HTML5 的模板和框架：
http://speckyboy.com/2010/04/16/15-useful-css3-and-html3-templates-and-frameworks/

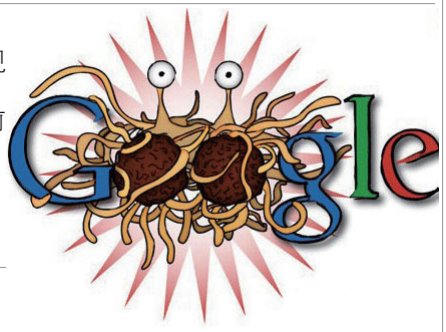
9)HTML 5 Web 开发人员的指南：
http://dev.w3.org/html5/html-author/

10)HTMLGoodies 的 HTML 基础：
http://www.htmlgoodies.com/primers/html/

11)HTML 5 下一代 Web 开发标准详解：
http://developer.51cto.com/art/200907/133407.htm

Google强推Dart语言替代JavaScript

我不理解为什么人们非要跟着 Google 帮他们用自己的技术去取代现有的技术。他们总是承诺“我们必须尝试新标准”，但他们自己的技术目前为止没有任何一样成为标准



先说 DART，在 10 月即将召开的 GOTO 开发者大会上，Google 工程师说他们将宣布一个新的编程语言的细节，而其名称就叫做 DART。即将在 GOTO 开发者大会上做主题演讲的是 Google 的工程师 Lars Bak，它负责 Chrome V8 Javascript 引擎，之前有过虚拟机的开发经验。Bak 持有大量专利，不过主要都来自早先在 SUN 工作时期获得的。

另外一位 Google 工程师 Bracha 则在博客里透露了一些内容，其先是表达了对 Javascript 的不满，然后提出互联网需要继续进化，另外就是需要能够利用上多核心处理器的性能。他还说：

Javascript 依然是一个对平台部署来说有着严重限制的语言。

然后神人出现了，挖到了去年 11 月 Google 员工之间的通信，自打那时候起就有一组 Google 团队在研究 Javascript 的未来了。这封信里提到的 Google 的新语言称为 Dash，看来现在已经改名为 Dart。但是这封信里的一些内容让人看了不禁后背发凉：

我们将严重鼓励 Google 开发者只专注于 Chrome，这样才可给我们带来最佳的用户体验。

那些仅专注于 Chrome 的开发者可以 xxxxx……专注于所有浏览器的开发者将不得不 xxxxx……

REDDIT 上有人评论到，我不理解为什么人们非要跟着 Google 帮他们用自己的技术去取代现有技术。他们总承诺“我们必须尝试新标准”，但他们的技术目前为止没有任何一样成为标准：

Google 不喜欢 MPEG，于是搞出了 VP8。但打一开始他们就没在将其打造成一个真正的开放标准上做任何努力。

Google 不喜欢 HTTP，于是搞出了 SPDY。但现在只有 Chrome 和 Google 的网页应用支持，目前也没有任何迹象能成进入标准组织。你可以想象微软的 IE 自己鼓捣了一套 HTTP 标准然后只和微软自己的 IIS 服务通讯吗？

Google 不喜欢所有非 IE 浏览器的 NPAPI 插件模型，于是搞了完全不兼容的插件 API 和原生代码的 Native Client。

Google 不喜欢 JPG 和 PNG，于是搞了 WebP 图形格式。

而现在 Google 又开始不喜欢 Javascript 了，于是搞了 Dart。

估计接下来 Google 还会不喜欢 CSS 甚至是 HTML。■

本文未完，查看详细请点击：

http://developer.51cto.com/art/201109/290663.htm

什么是Node.js?

Node 不是万能药！但的确能解决一些关键问题

学习 Node 不是一件轻松事儿，但你所收到的回报是对得起你的付出的。因为当下 Web 应用开发中的诸多难题唯有 JavaScript 才能解决。

- 专家们的警告！
- Node：几个小例子
- Node 不是 JavaScript，Node 可以运行 JavaScript
- 和 Node 服务器的交互
- 快速入门手册
- 解释器之惑
- 基于事件的 Web 应用
- Node 的用武之地

“你够酷吗？来用我吧！” Node.js 为最新潮的编程语言提供了一系列很酷的 API 和工具箱，它可以直接应用于传统的 Rails、Ajax、Hadoop、甚至可以某种程度上用于 iPhone 开发和 HTML5。如果你参加过一些大型技术会议，你总是会听到一些关于 Node.js 的主题演讲，尽管这些话题对普通的开发者来说依然有些难以企及。

你可能已经听说 Node.js（有时我们将其简称为“Node”）是一个服务器端的解决方案，它可以运行 JavaScript，并可以作为 Web 服务来处理 HTTP 请求。如果这些东东还不至于让你晕头转向的话，转眼间关于端口、sockets 和线程的讨论就又成了当下最热门的话题，你会觉得这些东西让你眼花缭乱。这些内容真的属于 JavaScript 的范畴吗？为什么世界上那么多人宁愿将 JavaScript 脱离浏览器而运行，更不用说将 JavaScript 运行于服务器端了？

好消息是，你所听到的（所想到的）关于 Node 的一切都是正确的。Node 的的确确是属于网络编程的范畴，用以处理服务器端的请求和响应。坏消息是和之前的 Rails、Ajax 和 Hadoop 一样，真正实用的技术资料实在太少。等到基于 Node 的“优秀的”框架成熟之后，技术资料一定会跟得上的，但何必要等到技术书籍和教程都出来之后再去尝试使用 Node 呢？现在就使用 Node，说不定会给你的代码带来意想不到的改观，甚至让你的程序变得更易实现。

专家们的警告！

和大多数技术一样，Node 也是新瓶装旧酒：它看起来不透明而且很怪异，但独受小开发团队的青睐。如果你没有接触过 Node，则需要学习一些很容易上手的服务器端脚本。你需要化时间来搞清楚 Node，因为即便是运行于服务器端的 JavaScript，它和客户端 JavaScript 也极为不同。实际情况是，你不得不自己给自己洗脑，以便重新学习理解围绕 JavaScript 的事件处理机制、异步 IO 和一些网络基础知识。

不幸的是，这意味着如果你已经用 Node 作开发超过两年时间的话，你会觉得这篇文章内容很单调乏而且过于简单。你会开始寻找新的“刺激”，比如将 Node 运行于客户端，或者开始尝试事件 I/O、反射器模式和 npm。你会发现 Node 的世界是如此有趣，甚至很多 Node 高级技术具有某种史诗般的美感，而这些东西对于初学者来说依然是难于企及的。

什么是 Node.js？ II

Node：几个小例子

首先，你应当意识到 Node 是用于运行独立的 JavaScript 程序的，而不是运行于浏览器中的某个 HTML 片段里。它是存放在文件系统中的真实存在的文件，由 Node 程序执行，以一种守护进程的模式运行，同时打开对某些端口的监听。

跳过 hello world

最经典的例子当然是“Hello World “，在 Node 官网 (http://nodejs.org/docs/latest) 上有源码。几乎每个人都是从 Hello World 开始接触 Node 的。现在让我们跳过这个最简单的例子，来看一些更有趣的例子：实现一个可以从服务器发送文件到客户端的程序 (而不仅仅是发送一段文本到客户端)。

```
var sys = require("sys"),
http = require("http"),
url = require("url"),
path = require("path"),
fs = require("fs");
http.createServer(function(request, response) {
    var uri = url.parse(request.url).pathname;
    var filename = path.join(process.cwd(), uri);
    path.exists(filename, function(exists) {
        if(!exists) {
            response.writeHead(404, {"Content-Type": "text/plain"});
            response.end("404 Not Found\n");
            return;
        }
        fs.readFile(filename, "binary", function(err, file) {
            if(err) {
                response.writeHead(500, {"Content-Type": "text/plain"});
                response.end(err + "\n");
                return;
            }
            response.writeHead(200);
            response.end(file, "binary");
        });
    });
}).listen(8080);
console.log("Server running at http://localhost:8080/");
```

感谢 Mike Amundsen，他给出了这段代码的相似的实现。这个例子是由 Devon Govett 在 Nettuts+ 上提交的一段代码，尽管已经根据新版本的 Node 作了更新，但 Devon 的整个帖子是一个非常好的入门学习教材，适合初学者。

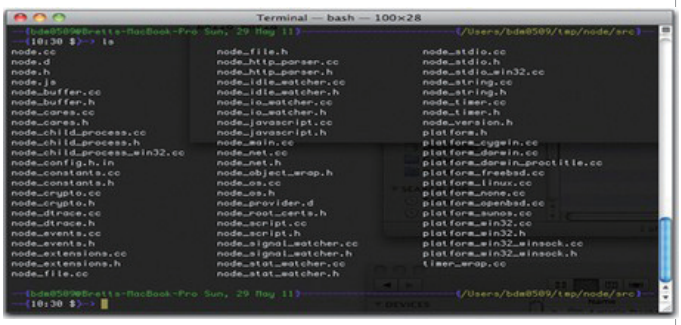
如果你是一个新手，你可以将上述代码保存到一个文本文件中，命名为 NodeFileServer.js。在运行之前你需要一个 Node 运行环境，最新的 Node 版本可以从官网下载这个文件或者从 github 上将源码取下来。你需要编译源码，如果你没有用过 Unix、对 make 和 configure 不甚熟悉，则需要查阅在线编译手册来寻求帮助。

Node 不是 JavaScript，Node 可以运行 JavaScript

刚刚你将 NodeFileServer.js 存成了某个文件，别担心，我们等下会回过头来运行它的。现在，让我们移步到现实当中来，在 Unix 中执行典型的配置和编译命令：

```
./configure
make
make install
```

这让我们确信一个事实：Node 不是 JavaScript，Node 是一个可以运行 JavaScript 的程序，但 Node 绝对不是 JavaScript。实际上，Node 是基于 C 写的程序。可以通过 ls 来查看 Node/src 目录中的文件，可以看到 Node 的源码：



（本文作者：李晶）本文未完，更多请点击：
http://developer.51cto.com/art/201109/288849_1.htm ■

大象的崛起！Hadoop七年发展风雨录

在互联网这个领域一直有这样的说法："如果老二无法战胜老大，那么就把老大赖以生存的东西开源吧"。

在互联网这个领域一直有这样的说法："如果老二无法战胜老大，那么就把老大赖以生存的东西开源吧"。当年 Yahoo! 与 Google 还是处在强烈竞 争关系时候，招聘了 Doug(Hadoop 创始人)，把 Google 老大赖以生存的 DFS 与 Map-Reduce 开源了，开始了 Hadoop 的童年时期。 差不多在 2008 年的时候，Hadoop 才算逐渐成熟。

从初创到现在，Hadoop 经过了至少 7 年的积累，现在的 Hadoop 不仅是当 年的老二 Yahoo 的专用产品了，从 Hadoop 长长的用户名单中，可以看到 Facebook、Linkedin、Amazon，可以看到 EMC、 eBay、Twitter、IBM、Microsoft,、Apple、HP... 国内的公司有淘宝、百度等等。

本文将对 Hadoop 七年(2004-2011)的发展历程进行梳理。读完本文后，将不难看出，Hadoop 的发展基本上经历了这样一个过程：从一个开源 的 Apache 基金会项目，随着越来越多的用户的加入，不断地使用、贡献和完善，形成一个强大的生态系统，从 2009 年开始，随着云计算和大数据的发 展，Hadoop 作为海量数据分析的最佳解决方案，开始受到许多 IT 厂商的关注，从而出现了许多 Hadoop 的商业版以及支持 Hadoop 的产品，包括软 件和硬件。

2004 年，Google 发表论文，向全世界介绍了 MapReduce。

2005 年初，为了支持 Nutch 搜索引擎项目，Nutch 的开发者基于 Google 发布的 MapReduce 报告，在 Nutch 上开发了一个可工作的 MapReduce 应用。

2005 年年中，所有主要的 Nutch 算法被移植到使用 MapReduce 和 NDFS(Nutch Distributed File System)来运行。

2006 年 1 月，Doug Cutting 加入雅虎，Yahoo! 提供一个专门的团队和资源将 Hadoop 发展成一个可在网络上运行的系统。

2006 年 2 月，Apache Hadoop 项目正式启动以支持 MapReduce 和 HDFS 的独立发展。

2007 年，百度开始使用 Hadoop 做离线处理，目前差不多 80% 的 Hadoop 集群用作日志处理。

2007 年，中国移动开始在"大云"研究中使用 Hadoop 技术，规模超过 1000 台。

2008 年，淘宝开始投入研究基于 Hadoop 的

大象的崛起！ Hadoop 七年发展风雨录 II

据。云梯1的总容量大概为9.3PB，包含了1100台机器，每天处理约18000道作业，扫描500TB数据。

2008 年 1 月，Hadoop 成为 Apache 顶级项目。

2008 年 2 月，Yahoo! 宣布其搜索引擎产品部署在一个拥有 1 万个内核的 Hadoop 集群上。

2008 年 7 月，Hadoop 打破 1TB 数据排序基准测试记录。Yahoo! 的一个 Hadoop 集群用 209 秒完成 1TB 数据的排序，比上一年的纪录保持者保持的 297 秒快了将近 90 秒。

2009 年 3 月，Cloudera 推出 CDH(Cloudera's Distribution including Apache Hadoop) 平台，完全由开放源码软件组成，目前已经进入第 3 版。

2009 年 5 月，Yahoo 的团队使用 Hadoop 对 1 TB 的数据进行排序只花了 62 秒时间。

2009 年 7 月，Hadoop Core 项目更名为 Hadoop Common;

2009 年 7 月，MapReduce 和 Hadoop Distributed File System (HDFS) 成为 Hadoop 项目的独立子项目。

2009 年 7 月，Avro 和 Chukwa 成为 Hadoop 新的子项目。

2010 年 5 月，Avro 脱离 Hadoop 项目，成为 Apache 顶级项目。

2010 年 5 月，HBase 脱离 Hadoop 项目，成为 Apache 顶级项目。

2010 年 5 月，IBM 提供了基于 Hadoop 的大数据分析软件——InfoSphere BigInsights，包括基础版和企业版。

2010 年 9 月，Hive(Facebook) 脱离 Hadoop，成为 Apache 顶级项目。

2010 年 9 月，Pig 脱离 Hadoop，成为 Apache 顶级项目。

2011 年 1 月，ZooKeeper 脱离 Hadoop，成为 Apache 顶级项目。

2011 年 3 月，Apache Hadoop 获得 Media Guardian Innovation Awards 。

2011 年 3 月， Platform Computing 宣布在它的 Symphony 软件中支持 Hadoop MapReduce API。

2011 年 5 月，Mapr Technologies 公司推出分布式文件系统和 MapReduce 引擎——MapR Distribution for Apache Hadoop。

2011 年 5 月，HCatalog 1.0 发布。该项目由 Hortonworks 在 2010 年 3 月份提出，主要用于解决 HDFS 存储瓶颈。

2011 年 4 月，SGI(Silicon Graphics International) 基于 SGI Rackable 和 CloudRack 服务器产品线提供 Hadoop 优化的解决方案。

2011 年 5 月，EMC 为客户推出一种新的基于开源 Hadoop 解决方案的数据中心设备，以助其满足客户日益增长的数据分析需求并加快利用开源数据分析软件。EMC 将通过一个基于分布式的 Hadoop 解决方案集成自己的 Greenplum 软件，从而可以在一个可扩展的设备里进行海量数据分析任务 (GFS+Greenplum)。Greenplum 是 EMC 在 2010 年 7 月收购的一家开源数据仓库公司。

更多内容，请参看：■

http://database.51cto.com/art/201109/290319.htm

SQL Server全文索引的硬伤

想象这样一个场景：在 DataBase_name.dbo.Table_name 中有一个名为 Title(标题) 和 Contents(内容) 的字段，现在需要查询在 Title 或者 Contents 中包括 “qq” 字符的所有记录。

面对这样的一个场景，我们通常都会写这样一个脚本：SELECT * FROM DataBase_name.dbo.Table_name WHERE Title LIKE '%qq%' OR Contents LIKE '%qq%'; 没错，这也是我第一个想到的方法。

但是我们需要思考的是：随着时间的推移，数据会越来越大，那个时候我们该如何提高我们的性能？客户随时都有可能要求加入对 Remark(备注) 字段的查询，难道我们就应该不厌其烦地修改程序代码？

面对上面的质问，我们需要提醒你的是：①对于这样的查询条件，即使 Title 和 Contents 上都有索引，我们也无法使用到索引，因为在 '%qq%' 的 “qq” 前面使用了通配符，所以无法使用到索引；如果查询的条件是 'qq%'，那倒是可以利用上索引。②在许多数据库性能调优的文章上都说 OR 这个谓词可以使用 SELECT UNION ALL SELECT 这样的方式来提高性能，但是需要提醒大家的是：如果在一条记录中字段 Title 和 Contents 都同时存在 “中国” 字符的话，那么返回的结果就会出现两条相同的记录，如果你希望是唯一的记录，那么这个时候你就要注意了。③其实有些时候，对于 and 的操作符，我们可以考虑使用：SQL Server 索引中 include 的魅力 (具有包含性列的索引)

现在回到我们上面提出的疑问上，大概这个时候大家都应该想到了数据库的全文索引了。全文索引是一种特殊类型的基于标记的功能性索引，由 Microsoft SQL Server 全文引擎 (MSFTESQL) 服务创建和维护。创建全文索引的过程与创建其他类型的索引的过程差别很大。MSFTESQL 不是基于某一特定行中存储的值来构造 B 树结构，而是基于要索引的文本中的各个标记来创建倒排、堆积且压缩的索引结构。(摘自 MSDN)

讲了那么久，硬伤在哪里呢？可能大家都怀疑我是不是标题党了，呵呵，马上就讲到，那就是这个全文索引能解决我们一开始提到的场景吗？回答是否定。为什么呢？因为 SQL Server 对字符串 “tqq.tencent.com” 进行分词和倒排索引后，我们是无法通过查询条件 ‘“*qq*”’ 来返回上面那条字符串的记录的，这样的查询条件只能查询到类似 “qqt.tencent.com”、“www.qq.com” 这样的字符串。SQL Server 的分词应该是正向最大值的分词方法，它没有把字符串进行反方向再进行一次分词和索引，所以只能查询到词或短语的前缀符合的记录。这一点有可能会被大家所忽略掉。

就针对上面的说法，我们来进行测试一下：

详细测试请参看：http://database.51cto.com/art/201108/287978.htm ■

编者按

索引是对数据库表中一列或多列的值进行排序的一种结构，使用索引可快速访问数据库表中的特定信息。除了数据库索引之外，在 LAMP 结果如此流行的今天，数据库性能优化也是海量数据处理的一个热点。

海量数据处理之数据库索引及优化

索引是对数据库表中一列或多列的值进行排序的一种结构，使用索引可快速访问数据库表中的特定信息。

数据库索引

什么是索引

数据库索引好比是一本书前面的目录，能加快数据库的查询速度。

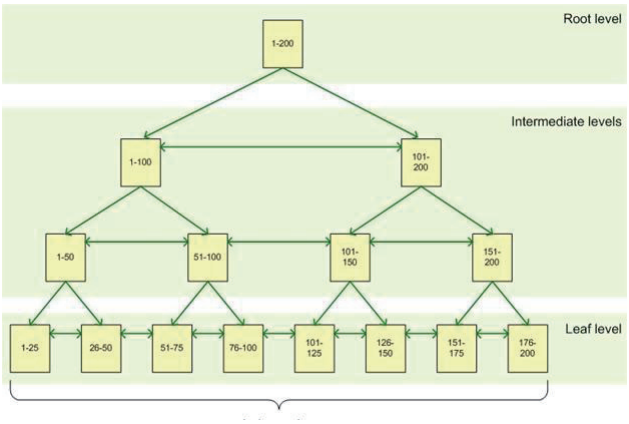
例如这样一个查询：select * from table1 where id=44。如果没有索引，必须遍历整个表，直到 ID 等于 44 的这一行被找到为止；有了索引之后（必须是在 ID 这一列上建立的索引），直接在索引里面找 44（也就是在 ID 这一列找），就可以得知这一行的位置，也就是找到了这一行。可见，索引是用来定位的。

索引分为聚簇索引和非聚簇索引两种，聚簇索引是按照数据存放的物理位置为顺序的，而非聚簇索引就不一样了；聚簇索引能提高多行检索的速度，而非聚簇索引对于单行的检索很快。

概述

建立索引的目的是加快对表中记录的查找或排序。

为表设置索引要付出代价的：一是增加了数据库的存储空间，二是在插入和修改数据时要花费较多的时间（因为索引也要随之变动）。



B 树索引 –Sql Server 索引方式

为什么要创建索引

创建索引可以大大提高系统的性能。

第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

第二，可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义

第四，在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间

第五，通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

也许会有人要问：增加索引有如此多的优点，为什么不对表中的每一个列创建一个索引呢？因为，增加索引也有许多不利的方面。

海量数据处理之数据库索引及优化 II

第一，创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

第二，索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。

第三，当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。

在哪建索引

索引是建立在数据库表中的某些列的上面。在创建索引的时候，应该考虑在哪些列上可以创建索引，在哪些列上不能创建索引。一般来说，应该在这些列上创建索引：

在经常需要搜索的列上，可以加快搜索的速度；

在作为主键的列上，强制该列的唯一性和组织表中数据的排列结构；

在经常用在连接的列上，这些列主要是一些外键，可以加快连接的速度；在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的；

在经常需要排序的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间；

在经常使用在 WHERE 子句中的列上面创建索引，加快条件的判断速度。

同样，对于有些列不应该创建索引。一般来说，不应该创建索引的的这些列具有下列特点：

第一，对于那些在查询中很少使用或者参考的列不应该创建索引。这是因为，既然这些列很少使用到，因此有索引或者无索引，并不能提高查询速度。相反，由于增加了索引，反而降低了系统的维护速度和增大了空间需求。

第二，对于那些只有很少数据值的列也不应该增加索引。这是因为，由于这些列的取值很少，例如人事表的性别列，在查询的结果中，结果集的数据行占了表中数据行的很大比例，即需要在表中搜索的数据行的比例很大。增加索引，并不能明显加快检索速度。

第三，对于那些定义为 text, image 和 bit 数据类型的列不应该增加索引。这是因为，这些列的数据量要么相当大，要么取值很少，不利于使用索引。

第四，当修改性能远远大于检索性能时，不应该创建索引。这是因为，修改性能和检索性能是互相矛盾的。

当增加索引时，会提高检索性能，但是会降低修改性能。当减少索引时，会提高修改性能，降低检索性能。因此，当修改操作远远多于检索操作时，不应该创建索引。

数据库优化

此外，除了数据库索引之外，在 LAMP 结果如此流行的今天，数据库 (尤其是 MySQL) 性能优化也是海量数据处理的一个热点。

下面就结合自己的经验，聊一聊 MySQL 数据库优化的几个方面。

首先，在数据库设计的时候，要能够充分的利用索引带来的性能提升，至于如何建立索引，建立什么样的索引，在哪些字段上建立索引，上面已经讲的很清楚了，这里不在赘述。■

本文未完，更多内容，请看：

http://database.51cto.com/art/201108/285011.htm

《移动互联网》第一季终于hold住了

就美剧的风格而言，已经没有更好的噱头来留住观众、留住收视率了，需要一个漂亮而耐人深思的结尾了。“乔帮主”辞职的消息，整个局面终于被乔布斯 hold 住了。

第一季结束了

终于结束了。

最近的移动新闻就像紧张美剧一样，一集一集环环相扣，在本季的最后一结，曾经在移动顶端的主人公“乔帮主”，自己终结了自己。究竟年迈、体弱的“乔帮主”在下季能否复出，没有人能给出答案。这个“美剧”，观众、导演说了都不算，得看乔布斯自己的意思。

嗅觉灵敏的媒体这次栽了。因为，每一条新闻都足以当版面的头条，就在各大媒体准备花时间深度挖掘新闻的时候，震惊的消息又来了……这几天说了多少个又了。

从 2008 年 11 月开始，一场漫长的移动互联网“专利罗圈架”打到现在还没有收手。这只不过是本剧的背景，时势造英雄，编剧们自然不会犯这个错误。

8 月 15 日晚间，真是磨刀不误砍柴工，一直不言不语的谷歌在打“罗圈架”之余，瞬间就把摩托罗拉移动收编了。观众们本以为摩托罗拉是个闯荡多年的老江湖，就算打不过人家，也可以左右逢源不至于束手就擒，谁曾想……真是长江后浪推前浪，前浪死在沙滩上。

8 月 19 日，惠普放弃了 webOS 并剥离了 PC 业务，什么 Palm、平板都已经成为了浮云。TouchPad 99 美元的价格已经被抢疯了，之前不是许多人说 webOS 一无是处么？最近媒体上又出现了 webOS 如何流畅、如何高效的言论，这不是 * 么。价格决定一切，在它面前，媒体和消费者都是无力的。可惜国内买不到，可惜了。

不只是移动领域，惠普曾经引以为豪的 PC 业务也被剥离，改回小名 "Compaq” 也是有可能的。出身 SAP 的现惠普 CEO 经营思路就是霸气，只不过笑的人会是 DELL、联想和苹果罢了。

有人说放弃是一种美，有人说惠普在下一盘更大的棋，有人说又一个 IBM（IBM 把 PC 和笔记本事业部出售给联想）诞生了，有人说——惠普终于解脱。

就美剧的风格而言，已经没有更好的噱头来留住观众、留住收视率了，需要一个漂亮而耐人深思的结尾了。

《移动互联网》第一季终于 hold 住了 II

8 月 25 日，“乔帮主” 辞职。

整个局面终于被乔布斯 hold 住了。

相比这些 “剧情”，雷布斯的小米手机发布就成杯具了；前段时间媒体热炒的各品牌的 “云手机” 话题瞬间被抛弃，瞬间。相比主线剧情，相比大牌的主角们，配角无论戏份还是名气都还需要锤炼。

疯狂的剧情总是没有一点征兆，“高潮” 不断的背后与移动互联网业界有着千丝万缕的联系，偶然 or 必然？不知道，疯狂的编剧令我等观众望尘莫及。观众们已经无力深入思考每集背后的隐情，我们的大脑被冲刷的近乎凌乱。

第二季

这一季中的主角乔布斯给我们留下了无数美好的惊喜和回忆，因为乔布斯，《移动互联网》第一季的收视率得以保证。作为观众，我们也很幸运，因为乔布斯，第一季，很精彩。可惜，天下没有不散的筵席。

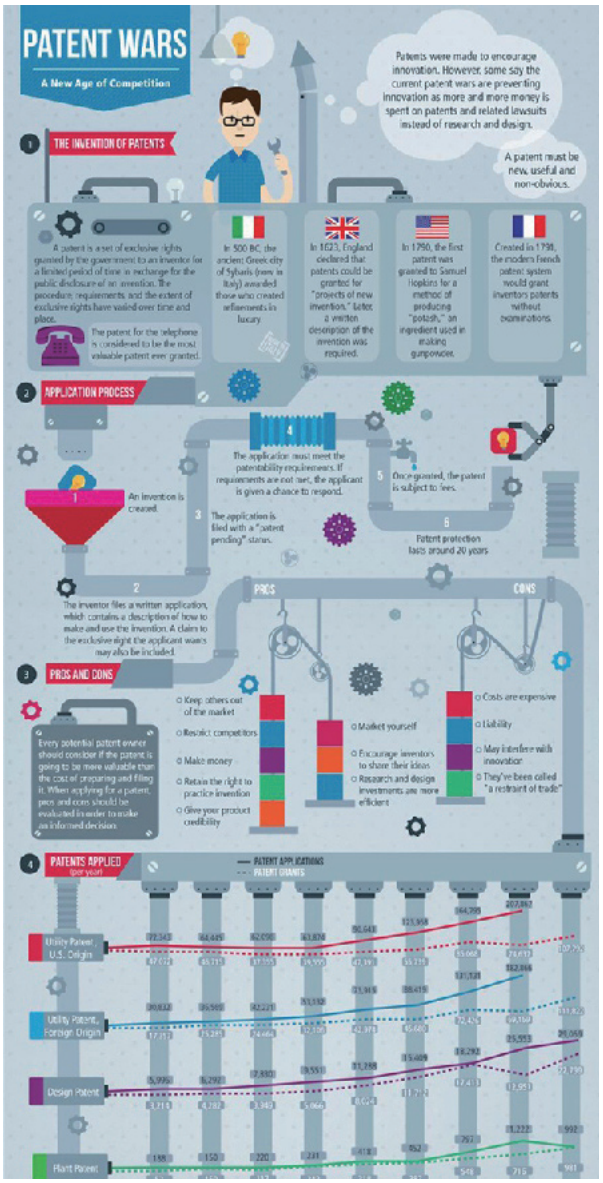
第二季是什么样子？这戏还好看么？谁来挑大梁？我们不得而知，但我们知道了上一季的大牌不在了，我们知道它们的 “罗圈架” 还没打完。

其实，“罗圈架” 并不能算作背景，它只是个导火索。真正的背景也许是美元信用评级的下降？也许 “乔帮主” 除了身体原因，因为看到了世界经济的不景气而功成身退；也许惠普放弃消费级领域业务，也是对社会消费潜力的不看好。也许谷歌收购摩托移动，只是一个初出茅庐的青年做的一个 “移动梦”。

也许，隐退的大牌并不只一个乔布斯，诺基亚、BlackBerry 的角色重要性也受到了严重的挑战。如果它们也 ** 了，剧本中的大牌真的一个手就能数过来了。

不管了，第二季上了再说。《移动互联网》第二季，究竟谁来 hold 住局面，我们拭目以待。

51CTO 最后给大家看一下第二季各位主人公 “罗圈架” 关系图吧（专利战）。



原文：

http://mobile.51cto.com/hot-286709.htm ■

Scala:潜力巨大的Android编程语言

静态类型 Java 语言以 JRuby 及 Groovy 的后备队的身份，在移动 Android 应用程序开发领域整装待发。但是、Scala 却未受 Android 开发者重视。

Scala，这款运行于 Java 虚拟机之上的静态类型语言，正逐渐成为谷歌 Android 应用程序开发领域的新选择。

尽管开发者对 Java 及 C++ 语言在 Android 上的使用已经非常熟悉，但他们可以选择采用其它 JVM 语言，因为 Android 一直在大力推动自己的虚拟机 Dalvik。工作于一家网页及移动应用程序开发公司的开发者 Mike Burns：“Scala 如今也具备了正在蓬勃发展的开发者社区”。

举例来说，来自波士顿 Android 开发组的成员们已经开始利用 Scala 进行 Android 开发，Burns 说道：“Scala 能够让大家更快地完成编码及分享的工作，尤其是与采用 Java 进行开发的过程相比。”作为该工作组负责人的 Burns 即将出席今年 10 月于旧金山举行的 Android 开放式会议，会上将讨论关于 Scala 替代 Java 的各项细节。

Scala 的创造者 Martin Odersky 进一步证实了 Burns 的主张：“我对他的说法深表赞同：Scala 是一款独具魅力的 Android 开发语言。另外值得一提的是，Android 插件中还具备对 SBT，即 Scala 标准创建工具的支持。”谷歌目前拒绝对 Scala 在 Android 开发领域的相关话题发表评论。但谷歌的项目托管站点上已经设有 Scala Android 页面，展示通过 Scala 为 Android 创建应用程序过程中所需的各类工具。

对于移动设备上的应用程序，Scala 比 JRuby、Groovy 等语言更具优势。因为静态类型语言运行速度更快、内存占用更少、更加优化。因为 Android 系统往往运行于处理器速度缓慢且内存较小的嵌入式设备上。

Burns 曾在一篇博文中宣扬 Scala 的编程特点：“Scala 可以被看作 Java 的改进版。首先，大家不再需要输入大量分号。Scala 带给我们的是现代抽象科学的创造力。添加属性、固有内容、值的类型检测以及信息块——所需的每种要素如今都以实实在在的编码成品方式摆在我们面前。”

分析师 Jeffrey Hammond 表示：“Dalvik 虚拟机使用自身的字节码格式来执行应用程序。谷歌将 Java 作为媒介语言以对 .dex 格式的文件进行编辑，Mike 似乎在寻找一种方式，旨在将 Scala 作为编写应用程序以及生成 .dex 编码类型文件的一种后备方案。”Burns 已经用 Scala 编写了一些 Android 应用，其中包括一款以 Umbrella Today 为基础的天气预报软件。他向我们列举了用 Scala 开发的优势，如速度快、编程容易以及充满活力的相关社区等等。不过，缺点也确实客观存在，其中包括以实践经验匮乏及缺少文档资料。

原文：http://mobile.51cto.com/android-288175.htm ■

在这儿IM：很职业很SoLoMo的LBS应用

51CTO 近期对跨平台的 LBS 应用商“在这儿 IM”CEO 熊尚文进行了专访，51CTO 与您一起了解这款定位清晰的商务社交应用。

作者 / 立方

中秋放假前，接到“在这儿 IM”（以下简称在这儿）的 CEO 熊尚文邮件：“‘在这儿’上线 20 天用户已经超过一万。”这个成绩对于市场上多如牛毛的 LBS 应用来说并不算非常出色，但是就像一位风投所说：“‘在这儿’是一款不以泡妞为目的的 LBS 交友应用了！”

没错，这是一款挖掘职场人脉的 LBS 应用。我们都身处商业社会，人脉就是生产力的观点已被越来越多的人所认同。因此，即使工作再忙，很多人还是乐此不疲地出席各种会议、活动、论坛，目的无非是想拓展自己的人脉圈子，寻找更多的商机。51CTO 记者也曾参加过移动互联网的沙龙聚会，每次面对上百号人不知什么来路的潮男潮女，眩晕！这是笔者的第一感觉。一位英文远比中文讲的溜的香港人熊尚文，似乎也被这个问题困扰了很久，于是就有了 iOS 和 Android 跨平台的 LBS 应用“在这儿 IM”。上周 51CTO 记者对“在这儿”CEO 熊尚文进行了专访，让我们一起了解“在这儿”吧。

“在这儿 IM”CEO 熊尚文

用 LBS 发掘职业人脉

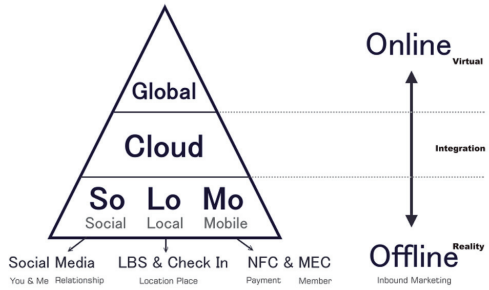
过去，人们基本都是通过交换纸质名片来获取对方的联系方式，以便进一步交流合作。随着科技的进步，“弹性社交”开始在美国流行，用户与其他用户并不拥有紧密的社会关系，通过弹性社交类产品可以进行交友、娱乐活动、商务交流等职业人脉交流。但是大家在“弹性社交”中最头疼的问题是缺乏“关于人的信息”。在聚会、论坛等这种人群密集的活动上，用户很难有针对性地直接找到感兴趣的人，可能换 10 张名片才能遇到一个真正感兴趣的人。所以出现了解决这种问题的应用——“你到会场，签到。有谁在，长相如何，干什么的能大概有个数。”

熊尚文向记者介绍：“在我们参加活动时我

在这儿 IM：很职业很 SoLoMo 的 LBS 应用 II

（接上页）们缺少对周围人职业的了解，我们可能把宝贵的时间浪费在无用的闲聊中，如何将职业人脉最大化地体现，是‘在这儿’的主要功能。”“在这儿”可以帮助用户发现周围有哪些人，然后用户可以选择跟感兴趣的人交换名片深入交流。

“在这儿”的 SoLoMo 解决方案



记者在询问熊尚文先生如何理解 LBS 业内最热的“SoLoMo”一词时，熊尚文是这样描述的“在这儿”的：“‘在这儿’有主要包括三大功能：活动聚集、交换电子名片、线上聊天。当我们需要参加活动时，活动聚集功能本身就是 Social 功能，在活动中我们基于同一相对较小的空间交换电子名片是就是 Local 功能。也就是说我们参加同一活动，在相互不认识的情况下交换了电子名片，经过线上聊天达成同一问题的共识、最后将距离抹平见面进行进一步沟通，达成 Mobile 的空间交流。可以说‘在这儿’就是一个 SoLoMo 的完美解决方案。”

◆活动聚焦

参加各种活动是扩展维系人脉的重要方式，“在这儿”会及时更新用户本地和周边的商务活动，便于用户查看选择。哪些人将会出席活动，他们的身份是什么，也一目了然，而且用户可以提前跟即将出席会议的人士交换名片。

◆交换电子名片

活动前可以交换名片，活动现场也无须挨个去打招呼，在线就能很方便的与感兴趣的人交换名片。同时“在这儿”能进行多重身份设定，你可以多重身份示人，这样一来，你不必担心因单一身份引起的我们分享、发表一些言论、话题时顾虑。“在这儿”支持建立多张电子名片，根据出席的场合决定你的名片。比如参加商务活动就使用职业名片，出席私人聚会时则用比较个性化的名片。

◆在线聊天

使用“在这儿”可以很轻松的在活动前和活动中获取有价值的人脉信息，但是频繁出席活动，难免会错过一些人，“在这儿”的在线聊天功能可以一定程度上所有弥补，活动结束后，用户依然可以跟现场没来得及打招呼的人建立联系，而且也可以通过在线聊天维系已经建立的联系，不需要再通过加 QQ 等方式，就能让你的人脉永葆活力。

后记

“在这儿 IM”是一款定位清晰的商务社交应用，解决的也是刚性需求，商业气息浓厚，对于这个小众市场的问题解决方案，就像作者在文章开头说的那样，能否争取到足够的用户还是未知数。但是对于创造“在这儿”的熊尚文和他的团队来说，从专业的领域出发，结合网络和智能手机的优势解决问题即为自然准确的 SoLoMo 解决方案。

对用户有用，是真正的价值。“在这儿”离已然成功！

原文：

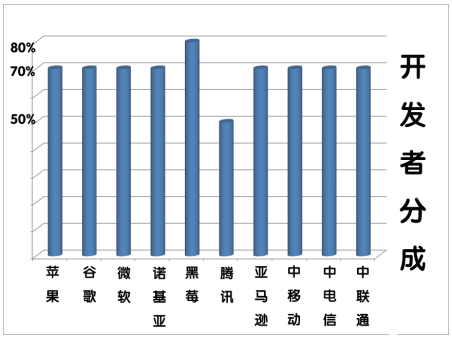
http://mobile.51cto.com/app-show-290624.htm ■

五五分成的腾讯移动应用商店

9月6日，腾讯正式发布了其基于移动互联网的应用平台腾讯应用中心，并邀请开发者和广告主加入其中。通过接入“腾讯应用中心”，开发者可以在web、wap、客户端等多个渠道推广自己的应用。 立方

目前各种应用商店已经多如牛毛。以苹果的app store为首，诺基亚、BlackBerry、三星等手机厂商都有自己的应用商店。移动、电信、联通三大电信运营商也纷纷推出了自己的在线软件商店。另外一些基于Android平台的小市场更是数不胜数。

腾讯应用中心的前身是手机腾讯网软件中心。2011年4月，手机腾讯网软件中心更名为腾讯应用中心。按照腾讯的思路，是希望通过“腾讯应用中心”打破原有移动互联网应用商店“各自为政”的商业模式，把开发者、广告主结合在一起，形成应用市场联盟，打造出一个应用生态圈，为用户提供各种应用。



腾讯从中向开发商收取推广费，或者通过腾讯的平台与开发者进行收入的“五五”分成。事实上这样的模式其实和一些移动互联网的广告联盟很相似，但腾讯应用中心商务总监童学红强调，腾讯来做这样的平台有一大好处就是，“我们至少在一两年没有盈利压力，可以将更多的营收分给开发者”。

51CTO短评：从上图我们可以看到，各个应用商店都非常依赖开发者，普遍与开发者的分成为“三七”分成，甚至有BlackBerry App World这样的小众应用商店与开发者“二八”分成，藉此来吸引开发者。参照腾讯近几年的扩张模式均以庞大的QQ用户群说事来打击竞争对手，但是如果腾讯对开发者实行“店大欺客”政策，不知道开发会不会买账呢？

Tips：什么是移动应用商店？

应用商店，是2009年由苹果公司提出的概念。应用商店诞生的初衷，是让智能手机用户在手机上完成更多的工作和娱乐。在09年底，手机软件商店的概念迅速风靡起来，各大手机厂商开始搭建自己的应用商店，来提升自身手机产品的卖点和吸引力。手机软件商店里的内容涵盖了手机软件，手机游戏，手机图片，手机主题，手机铃声，手机视频等几类。

最著名移动应用商店——App Store是苹果公司基于iPhone的软件应用商店，向iPhone的用户提供第三方的应用软件服务，这是苹果开创的一个让网络与手机相融合的新型经营模式。■

试用腾讯应用中心 感受QQ的微创新

本文为 51CTO 编辑对腾讯发布的移动应用商店——腾讯应用中心试用评测文章。腾讯依旧选择对业内已有产品的微创新，腾讯这次的目标是“创新工场的豌豆荚”。 立方

在腾讯发布基于移动互联网的应用平台（移动应用商店）后，51CTO 记者第一时间采访了一位同样做移动市场应用开发的朋友，他认为 QQ 应用商店能否成功还需时间的考验。朋友表示：“腾讯应用中心应该是 Android 应用商店。作为市场应用的开发者，我很关心腾讯应用商店的模式，是对亚马逊 Appstore 的微创新还是全新的应用模式呢？”

微创新一：没有网址的应用商店

51CTO 记者随后登陆腾讯应用中心，腾讯的手机应用平台有很详尽的上传应用流程，而腾讯应用中心却没有提供专门网页版移动应用商店，只提供了一个“应用助手 for Android”下载。记者在下载之后，发现这个商店应用 “联系人管理”、“ 短信收发”和“音乐视频管理”等功能均与“豌豆荚手机精灵”功能相同，应用助手显然可以看做是集成了“管理软件、商店与豌豆荚精灵的应用”。果不其然，腾讯依旧选择对业内已有产品的微创新，不过腾讯目标不是朋友预言的亚马逊，而是创新工场的豌豆荚。

微创新二：站在豌豆荚肩上的应用中心

随后 51CTO 记者用自己两款 Android 手机（摩托罗拉 MB525 和 ME501）测试了这款应用与豌豆荚不同之处。首先，笔者将两款手机连接电脑，腾讯应用中心（应用助手）的界面分别出现了两个手机的型号和系统版本号 (MOTO Defy 为 Android 2.3.4&MOTO ME501 为 Android 1.5)，并配有手机截图功能。同时还有 USB 和 WiFi 两种连接方式方便用户管理手机应用。

而笔者的 ME501 为未 Root 的机器，笔者试着使用应用助手删除手机 QQ，非常顺利！

微创新三：热点应用全是自家货

当然，最重要的是我们还要测试腾讯应用中心（应用助手）的商店功能。笔者用 Defy 下载了在 App Store 上下载量和排名已经超过“愤怒的小鸟”的“会说话的汤姆猫”，非常流畅，下载过后的自动安装速度奇快。不过当笔者查看“热点应用”时还是有些许不爽，整个版面大部分是 QQ 自家的应用，笔者只能安慰自己“腾讯应用中心推出不久，开发者上传的应用还不多”。

同时，应用助手在“通讯录管理”、“短信管理”和“照片录像管理”等功能的用户体验同样做得比“豌豆荚”要好很多。可见腾讯在研发这款应用时，在保证完成竞争对手应用功能的基础上一如既往地保持了其微创新的风格。笔者只是试用应用中心的部分功能，其他使用者在新浪微博上@51CTO 移动开发分享这款市场应用的新奇功能。 ■